KOOK W2

Continuous Optimization (I) unconstrained

EE, KAIST

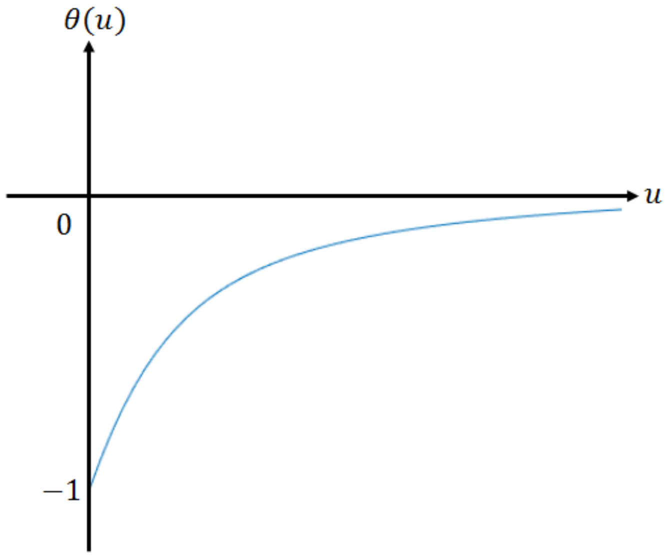
김창익

Contents

- Foundations
- Categorization
- Unconstrained optimization
  - Regression
  - Universally applied methods
  - Line search
  - Steepest descent method
  - Conjugate gradient method
  - Newton's method (and Quasi-Newton method)
  - Newton's method for nonlinear lease square problems
  - Lavenberg-Marquardt algorithm
- Reference

Foundations

- Infimum and supremum
  - Consider $\theta(u) = u - \sqrt{1+u^2}$
  - As u→ ∞, $\theta(u) \to 0.$
  - Hence, $\sup\{\theta(u) : u \geq 0\} = 0.$
  - But a maximizing solution u* does not exist.



Foundations

- Differentiability
  - Differential calculus is based on the idea of approximating an arbitrary function by an affine function.
  - A function $A : R^n \to R^m$ is affine if there exists a linear function $L : R^n \to R^m$ and a vector $\mathbf{y} \in R^m$ such that

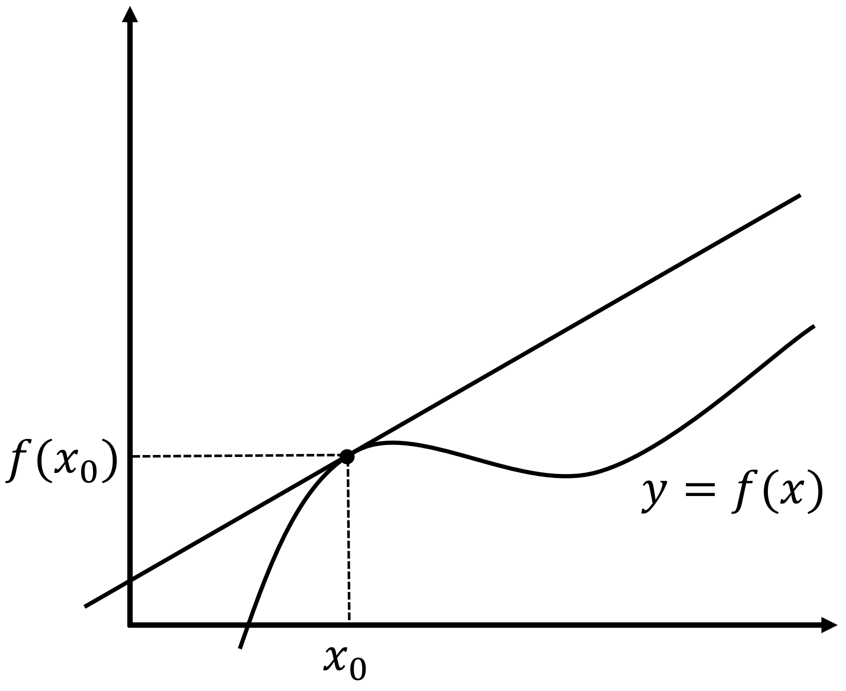
$$A(\boldsymbol{x}) = L(\boldsymbol{x}) + \boldsymbol{y}$$

for every $\boldsymbol{x} \in R^n$.

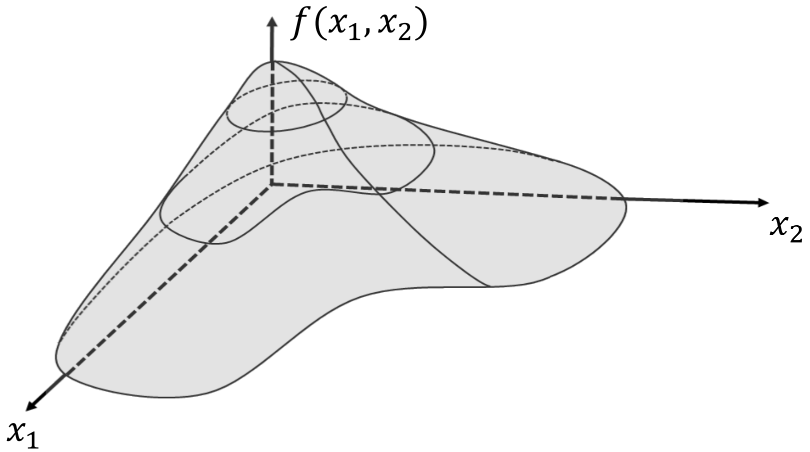
  - A function *f* is said to be differentiable at $\boldsymbol{x}_0$ if there is an affine function that approximates *f* near $\boldsymbol{x}_0$.
  - If $f : R^n \to R$ is differentiable, the gradient of *f* is defined

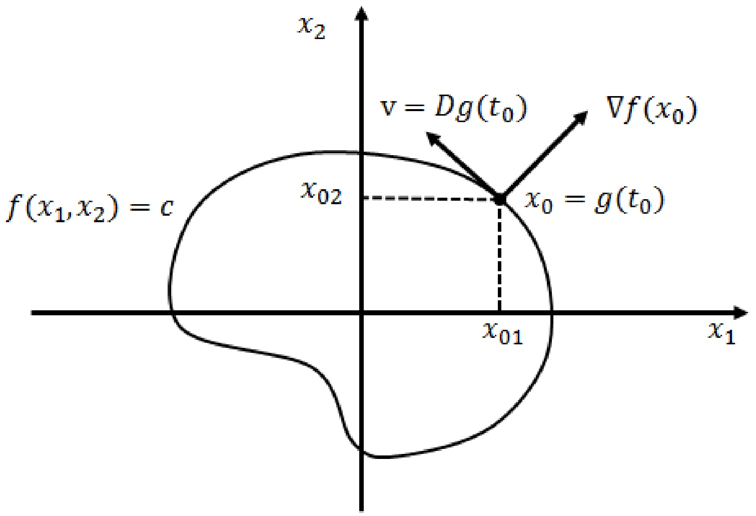
$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\boldsymbol{x}) \end{bmatrix} = Df(\boldsymbol{x})^T.$$

- Differentiability
  - Differential calculus is based on the idea of approximating an arbitrary function by an affine function.



- Level sets and gradients
  - The level set of a function $f: R^n \to R$ at level *c* is the set of points $S = \{\boldsymbol{x} : f(\boldsymbol{x}) = c\}$.
  - $\mathbf{v}$ is a tangent vector to curve *r* at $\boldsymbol{x}_0$
  - The curve r is parameterized by a continuously differentiable function $\boldsymbol{g}: R \to R^m$.
  - Since r lies on S, we have $f(\boldsymbol{g}(t)) = c$.
  - Thus, $Df(\boldsymbol{g}(t_0))D(\boldsymbol{g}(t_0)) = \nabla f(\boldsymbol{x}_0)^T \boldsymbol{v} = 0$



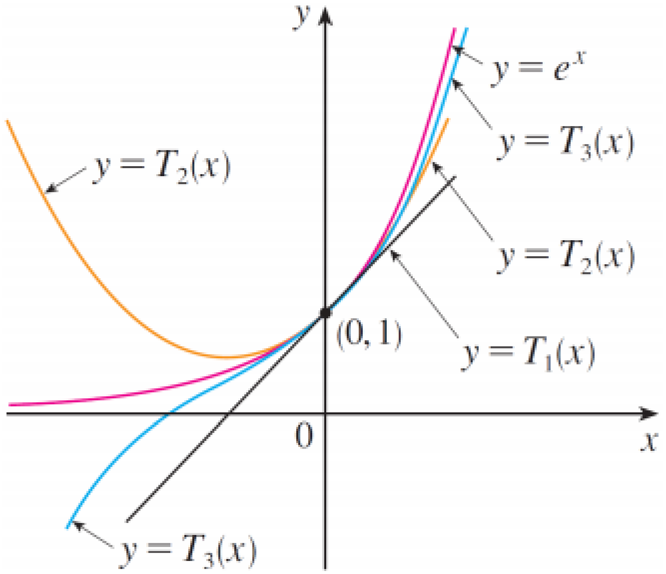


Foundations

- Taylor series

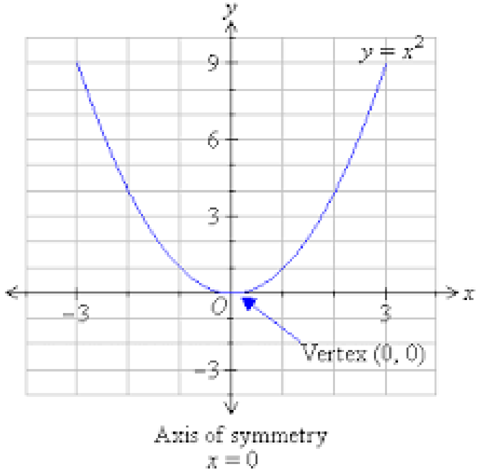
$$T_n(x) = \sum_{i=0}^{n} \frac{f^{(i)}(a)}{i!}(x-a)^i = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n$$

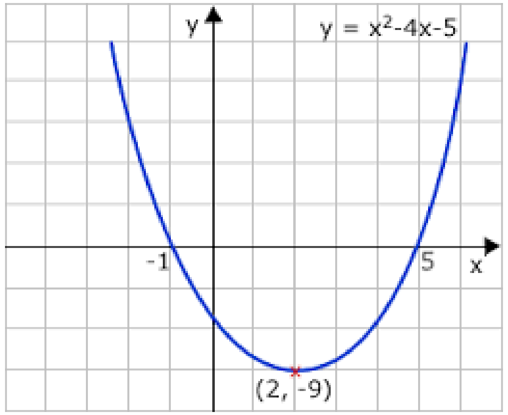
$$T_1(x) = 1 + x, \qquad T_2(x) = 1 + x + \frac{x^2}{2!}, \qquad T_3(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$



- Positive Semidefinite
  - Objective function is in the form of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} + d$ where $\mathbf{Q}$ is $n \times n$ symmetric matrix, $\mathbf{c} \in \mathbf{R}^n$, and $d \in R$.
  - Constraint functions are all affine.
  - If $\mathbf{Q}$ is <u>positive semidefinite</u>, $x^T Q x \geq 0$, then the problem is convex quadratic program.

$$ax^2 + bx + c = 0$$





Categorization

- Categorization of continuous optimization
  - Unconstrained optimization
    - ~~Linear~~
    - Nonlinear
  - Constrained optimization
    - Linear programming
    - Nonlinear programming

Categorization

- Unconstrained optimization:

$$\boldsymbol{x}^* = \min f(\boldsymbol{x})$$

$$\textit{subject to } \boldsymbol{x} \in \Omega,$$

$$\boxed{\text{where } \Omega = \boldsymbol{R}^n.}$$

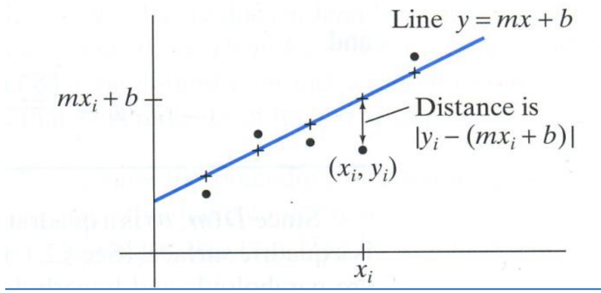
- Constrained optimization:
  - feasible set is the form of functional constraints.

$$\Omega = \{\boldsymbol{x} : \boldsymbol{h}(\boldsymbol{x}) = 0, \boldsymbol{g}(\boldsymbol{x}) \leq 0\}.$$

Unconstrained optimization(1)

Contents

Regression: Least squared method



$$\textit{minimize} \left\| \boldsymbol{Ax} - \boldsymbol{y} \right\|^2$$

- A special case where objective functions are in quadratic form:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q}\boldsymbol{x} - \boldsymbol{c}^T \boldsymbol{x} + d$$

  – The minimum can be found analytically by setting its derivative to zero.

$$\partial f(\boldsymbol{x}) / \partial \boldsymbol{x}$$

$$\boldsymbol{Q} \cdot \boldsymbol{x}^* = \boldsymbol{c}$$

Regression: Least squared method

- Example 1: Shading Correction (1/4)

  – Problem

    - Due to inhomogeneous illumination as well as optical effects, the observed intensity values often decrease at image borders, even if the camera image depicts a target of uniform reflectivity which should appear uniformly bright all over the image
    - The decline is approximated by a polynomial function $p_s(\boldsymbol{c}, \boldsymbol{x})$. Based on this polynomial, a correction function $l_s(\boldsymbol{x})$ can be derived such that $p_s(\boldsymbol{c}, \boldsymbol{x}) l_s(\boldsymbol{x}) = 1$ for every pixel position x .
    - It is assumed that a maximum order of four is sufficient for $p_s(\boldsymbol{c}, \boldsymbol{x})$ being a good approximation to observed shading.

$$
\begin{aligned}
p_S(\mathbf{c}, \mathbf{x}) = {} & c_0 + c_1 \cdot u^2 + c_2 \cdot v^2 + c_3 \cdot u^2 \cdot v^2 + c_4 \cdot u^4 + \\
& + c_5 \cdot v^4 + c_6 \cdot u^4 \cdot v^2 + c_7 \cdot u^2 \cdot v^4 + c_8 \cdot u^4 \cdot v^4
\end{aligned}
$$

where $u = |x - x_c|$ y and $v = |y - y_c|$

Regression: Least squared method

- Example 1: Shading Correction (2/4)

  – The polynomial equation

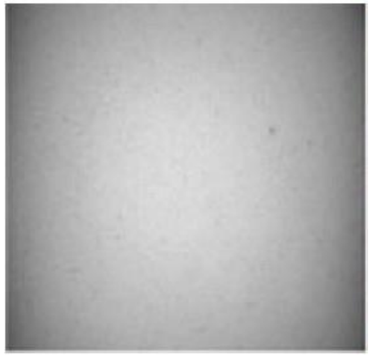
$$p_S(\mathbf{c},\mathbf{x}) = c_0 + c_1 \cdot u^2 + c_2 \cdot v^2 + c_3 \cdot u^2 \cdot v^2 + c_4 \cdot u^4 + \\ + c_5 \cdot v^4 + c_6 \cdot u^4 \cdot v^2 + c_7 \cdot u^2 \cdot v^4 + c_8 \cdot u^4 \cdot v^4$$

$u = |x - x_c|$ and $v = |y - y_c|$ Denote the pixel distances to the image center

  – Observed data I(x) (Calibration Image): Capture an image of a white board with uniform intensity

    - Maximum intensity normalization $\tilde{I}(\mathbf{x}) = \dfrac{I(\mathbf{x})}{I_{\max}}$

Calibration image



Uniform intensity image

Regression: Least squared method

- Example 1: Shading Correction (3/4)

Our goal is to find a parameter

$$\begin{bmatrix} 1 & u^2 & v^2 & \dots & u^4v^4 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_8 \end{bmatrix} = \bar{I}(\mathbf{x}) + e$$

$$\bar{I}(\mathbf{x}) = \frac{I(\mathbf{x})}{I_{\max}}$$

$$\begin{bmatrix} 1 & {u_1}^2 & {v_1}^2 & {u_1}^2{v_1}^2 & {u_1}^4 & {v_1}^4 & {u_1}^4{v_1}^2 & {u_1}^2{v_1}^4 & {u_1}^4{v_1}^4 \\ 1 & {u_2}^2 & {v_2}^2 & {u_2}^4{v_2}^4 & {u_2}^4 & {v_2}^4 & {u_2}^4{v_2}^2 & {u_2}^2{v_2}^4 & {u_2}^4{v_2}^4 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & {u_N}^2 & {v_N}^2 & {u_N}^2{v_N}^2 & {u_N}^4 & {v_N}^4 & {u_N}^4{v_N}^2 & {u_N}^2{v_N}^4 & {u_N}^4{v_N}^4 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_8 \end{bmatrix} \approx \begin{bmatrix} \bar{I}(\mathbf{x}_1) \\ \bar{I}(\mathbf{x}_2) \\ \vdots \\ \bar{I}(\mathbf{x}_N) \end{bmatrix}$$

$$\text{or } \mathbf{X} \cdot \mathbf{c} \approx \mathbf{d}$$

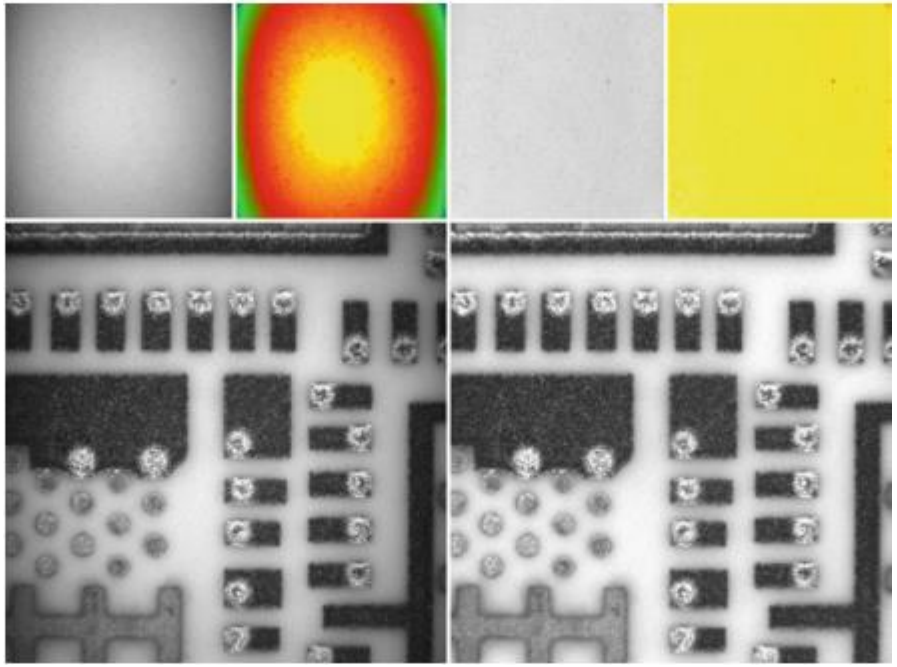
$$\mathbf{c}^* = \arg\min \|\mathbf{X} \cdot \mathbf{c} - \mathbf{d}\|^2$$

Regression: Least squared method

- Example 1: Shading Correction (4/4)

This minimizer c* of this quadratic form can be found by setting the derivative $\dfrac{\partial \|\boldsymbol{X} \cdot \boldsymbol{c} - \boldsymbol{d}\|^2}{\partial \boldsymbol{c}}$ to zero, which leads to

$$\boldsymbol{X}^T \cdot \boldsymbol{X} \cdot \boldsymbol{c}^* = \boldsymbol{X}^T \cdot \boldsymbol{d}$$

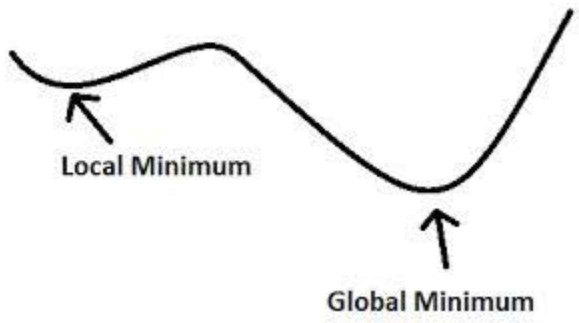


The system of normal equations can be solved by performing either a LR or QR decomposition of X^TX.

Another way is to calculate the pseudo-inverse of X^TX by performing SVD.

Universally applied methods

- Iterative multidimensional optimization:
  - Universally applied methods because they
    - Directly operate on the objective function $f(\boldsymbol{x})$
    - Do not rely on a special structure of $f(\boldsymbol{x})$
  - The other side of the coin is that these methods are usually rather slow.
  - Also typically perform a local search: global minimum not guaranteed

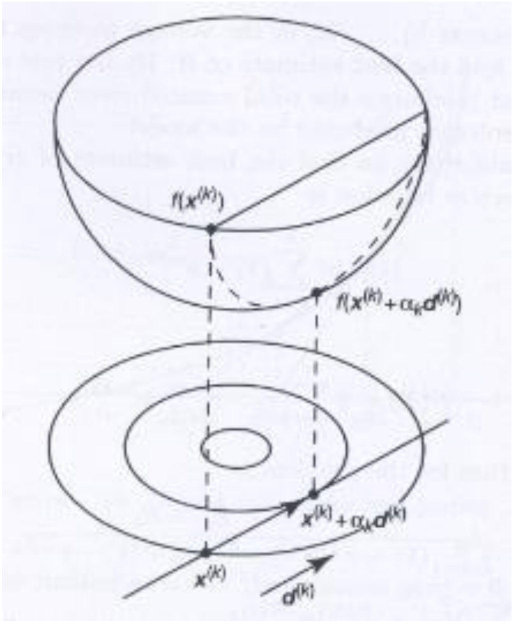


Universally applied methods

- Iterative multidimensional optimization:
  - Consists of two steps
    1. Calculation of a search direction $\boldsymbol{d}^k$ along which the minimal position is to be searched
    2. Update the solution by fining $\boldsymbol{x}^{k+1}$ which reduces $f(\boldsymbol{x}^{k+1})$ by performing a one-dimensional search along the direction $\boldsymbol{d}^k$. This is called a line search.

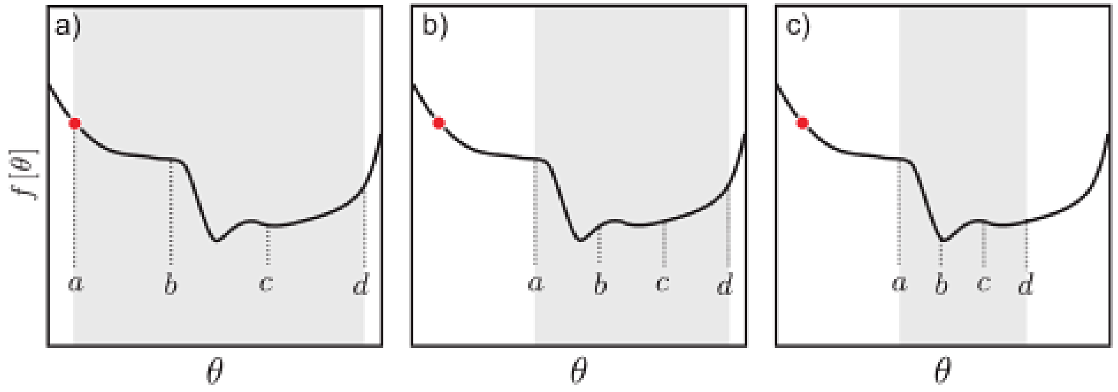
       (← next slide)
  - Mathematically

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$$
$$\alpha^k, \textit{step size}$$



Line Search

- One dimensional search along the search direction

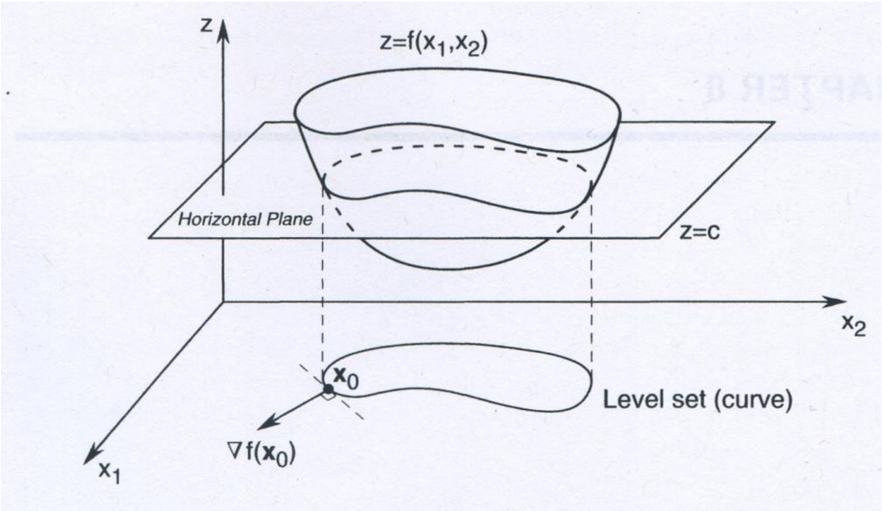


  – Golden section search, Fibonacci search, Newton's method….

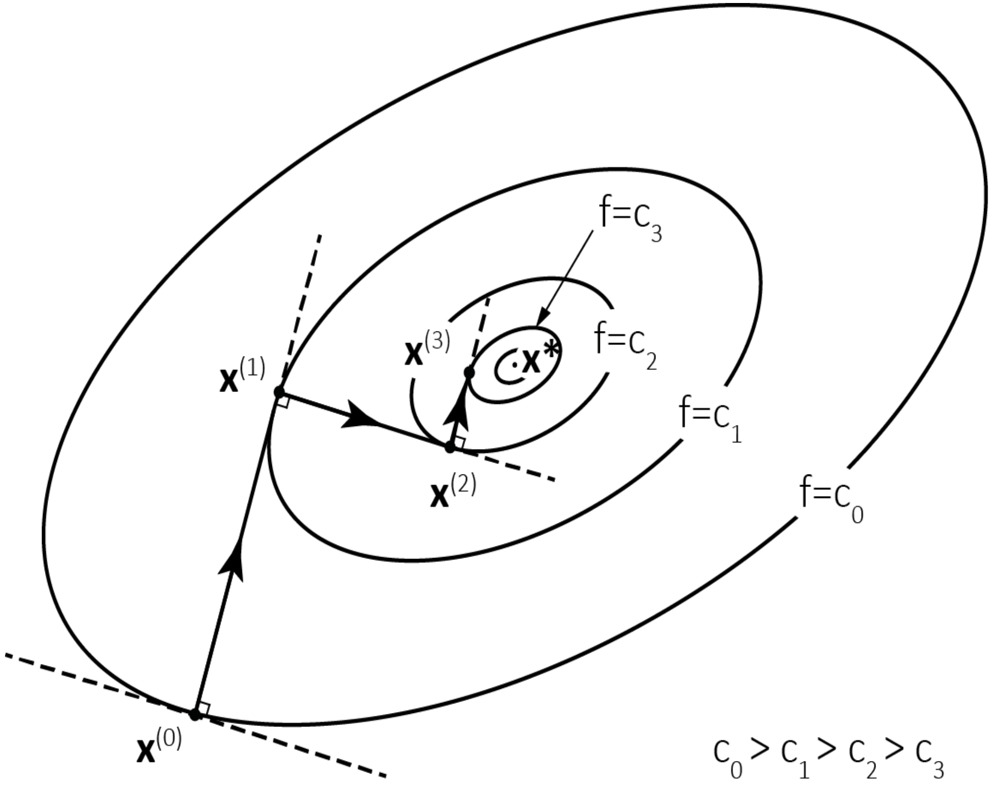
Steepest decent method

- The method of steepest decent is a gradient algorithm where the step size α_k is chosen to achieve the maximum amount of decrease of the objective function at each individual step.
  - Conduct two steps starting at x_0
    1. Find search direction : $-\nabla f(x_0)$
    2. Line search $\alpha_k = \arg\min_{\alpha \geq 0} f(x^{(k)} - \alpha \nabla f(x^{(k)})).$

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \alpha^k \boldsymbol{d}^k$$



Steepest decent method

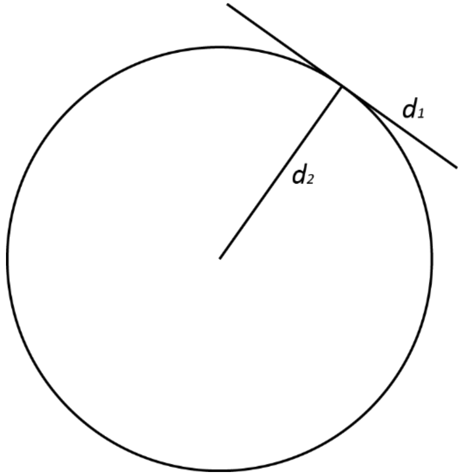


Conjugate gradient method (1951)

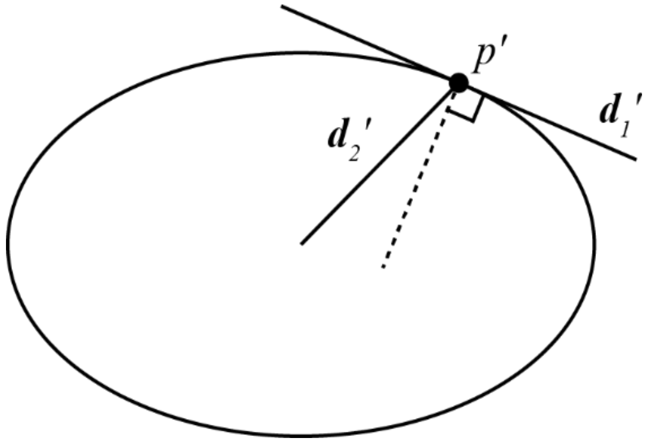
$$\boldsymbol{x} = [x_1, x_2]$$

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{x} = x_1^2 + x_2^2$$

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x}$$



$$x \to Gx$$



$$d_1 = G^{-1} d_1', d_2 = G^{-1} d_2'$$

$$d_1^T d_2 = (G^{-1} d_1')^T (G^{-1} d_2') = d_1'^T G^{-1T} G^{-1} d_2' = d_1'^T Q d_2' = 0$$

We say that the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are Q-conjugated (or Q-orthogonal if $\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{y} = 0$.

Conjugate gradient method

- Ex) For the given positive definite matrix Q, find the Q-conjugate directions

$$\boldsymbol{Q} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}$$

Let $\boldsymbol{d}^{(0)} = [1,0,0]^T$, since $\boldsymbol{d}^{(0)T}\boldsymbol{Q}\boldsymbol{d}^{(1)} = 0$

$$\boldsymbol{d}^{(0)T}\boldsymbol{Q}\boldsymbol{d}^{(1)} = [1,0,0]\begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}\begin{bmatrix} d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \end{bmatrix} = 3d_1^{(1)} + d_3^{(1)} = 0 \qquad \boldsymbol{d}^{(1)} = [1,0,-3]^T$$

To find third vector,

$$\left.\begin{aligned} \boldsymbol{d}^{(0)T}\boldsymbol{Q}\boldsymbol{d}^{(2)} &= 3d_1^{(2)} + d_3^{(2)} = 0, \\ \boldsymbol{d}^{(1)T}\boldsymbol{Q}\boldsymbol{d}^{(2)} &= -6d_1^{(2)} - 8d_3^{(2)} = 0, \end{aligned}\right\} \quad \text{We can take} \quad \boldsymbol{d}^{(2)} = [1,4,-3]^T$$

Conjugate gradient method

- The first search direction from an initial point $\boldsymbol{x}^{(0)}$ is in the direction of steepest decent, that is $\boldsymbol{d}^{(0)} = -\nabla f(\boldsymbol{x}^{(0)})$
- Then, new directions are calculated as a linear combination of the previous direction and the current gradient – in such a way that all the directions are mutually Q-conjugate

$$\boldsymbol{d}^{(k+1)} = -\nabla \boldsymbol{f}(\boldsymbol{x}^{(k+1)}) + \beta_k \boldsymbol{d}^{(k)}, \quad k = 0,1,2,\ldots$$

$$\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k+1)} = \boldsymbol{d}^{(k)T}\boldsymbol{Q}(-\nabla \boldsymbol{f}(\boldsymbol{x}^{(k+1)}) + \beta_k \boldsymbol{d}^{(k)}) = 0, \quad k = 0,1,2,\ldots$$

$$\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \alpha_0 \boldsymbol{d}^{(0)}$$

$$\alpha_0 = \arg\min_{\alpha \geq 0} f(x^{(0)} - \alpha \boldsymbol{d}^{(0)}) = -\frac{\nabla f(\boldsymbol{x}^{(0)})^T \boldsymbol{d}^{(0)}}{\boldsymbol{d}^{(0)T}\boldsymbol{Q}\boldsymbol{d}^{(0)}}$$ [Chong]

$$\beta_k == \frac{\nabla f(\boldsymbol{x}^{(k+1)})^T \boldsymbol{Q}\boldsymbol{d}^{(k)}}{\boldsymbol{d}^{(k)T}\boldsymbol{Q}\boldsymbol{d}^{(k)}}$$

Conjugate gradient method

- For nonquadratic problems, four of the best known formulas for β_n are named after their developers and are given by the following formulas:

$$\Delta \boldsymbol{x}_n = -\nabla \boldsymbol{f}\,(\boldsymbol{x}^{(n)})$$

- Fletcher–Reeves:

$$\beta_n^{FR} = \frac{\Delta x_n^\top \Delta x_n}{\Delta x_{n-1}^\top \Delta x_{n-1}}$$

- Polak–Ribière:

$$\beta_n^{PR} = \frac{\Delta x_n^\top (\Delta x_n - \Delta x_{n-1})}{\Delta x_{n-1}^\top \Delta x_{n-1}}$$

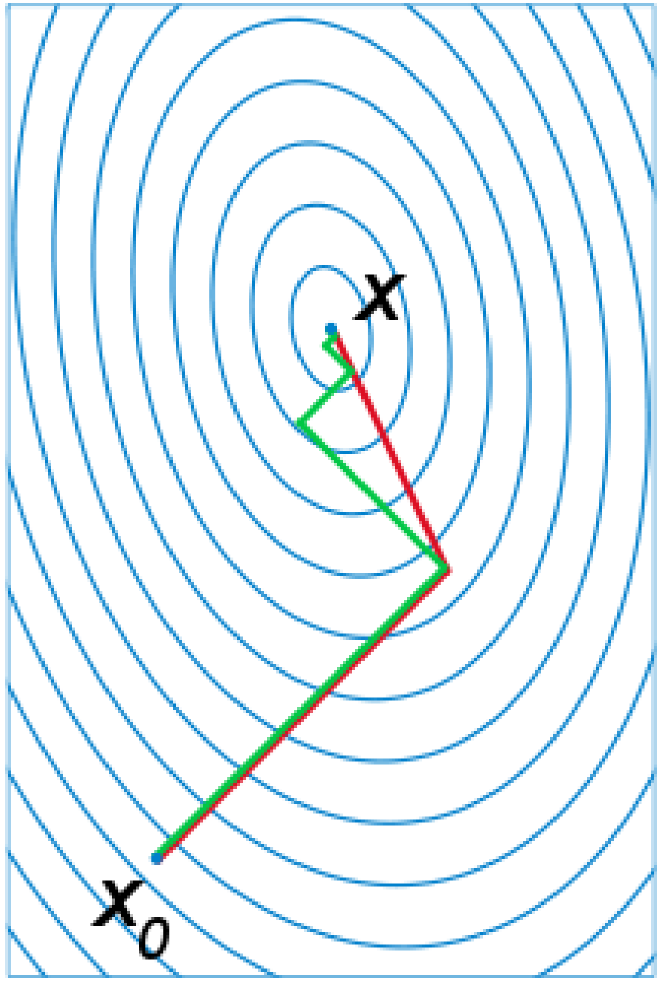
- Hestenes-Stiefel:

$$\beta_n^{HS} = -\frac{\Delta x_n^\top (\Delta x_n - \Delta x_{n-1})}{s_{n-1}^\top (\Delta x_n - \Delta x_{n-1})}$$

- Dai–Yuan:

$$\beta_n^{DY} = -\frac{\Delta x_n^\top \Delta x_n}{s_{n-1}^\top (\Delta x_n - \Delta x_{n-1})}.$$

Conjugate gradient method



A comparison of the convergence of steepest descent with optimal step size (in green) and conjugate vector (in red) for minimizing a quadratic function associated with a given linear system.

Conjugate Gradient is an intermediate between steepest descent and Newton's method.

It has proved to be extremely effective in dealing with general objective functions and is considered among the best general purpose methods presently available.

Unconstrained optimization(2)

Contents

Newton's method

- Recall that the steepest descent method uses only the first derivatives (gradients) in selecting a suitable search direction, which is not always the most effective.

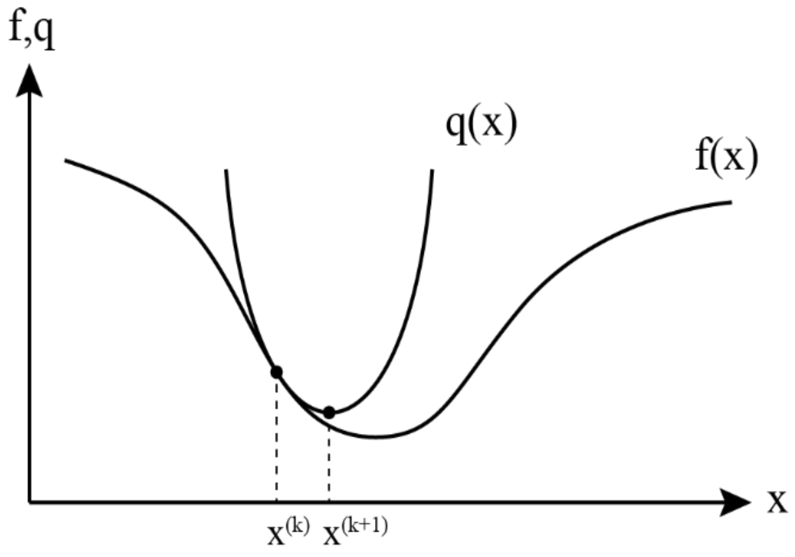
- Newton's method (also called the Newton-Raphson method) uses first and second derivatives and indeed does perform better than the steepest descent method.

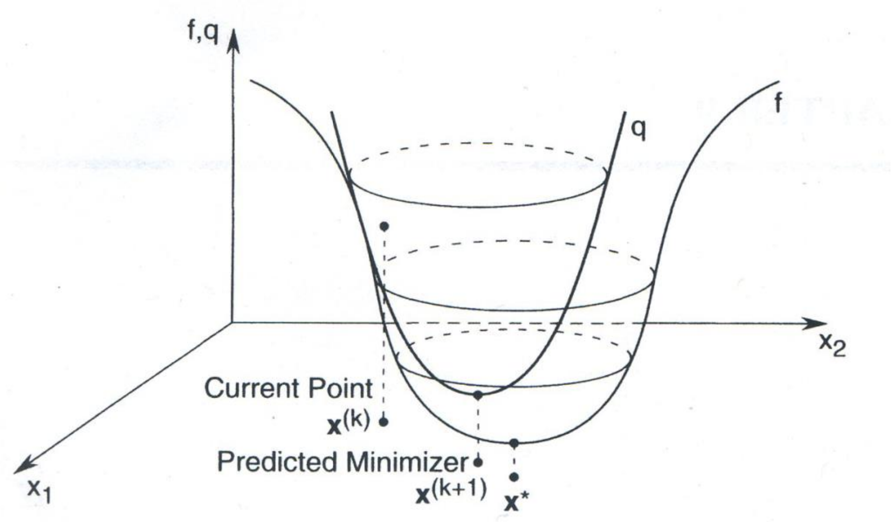
- The basic idea is to construct, given a starting point, a quadratic approximation to the objective function then minimize the approximate function instead of the original objective function. The minimizer becomes the starting point in the next step.

Newton's method

- Quadratic approximation using the Taylor series expansion of f(x) at the current point $\boldsymbol{x}^{(k)}$

$$f(\boldsymbol{x}) \approx q(\boldsymbol{x}) = f(\boldsymbol{x}^{(k)}) + (\boldsymbol{x} - \boldsymbol{x}^{(k)})^T \nabla f(\boldsymbol{x}^{(k)}) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^{(k)})^T H(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)})$$





Newton's method

- To find the minimizer of q(x)

$$\nabla q(\boldsymbol{x}) = \nabla f(\boldsymbol{x}^{(k)}) + H(\boldsymbol{x}^{(k)})(\boldsymbol{x} - \boldsymbol{x}^{(k)}) = 0.$$

- If $H(x^{(k)}) > 0$, then $q(\boldsymbol{x})$ achieves a minimum at $\boldsymbol{x}^{(k+1)}$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - H(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$$

This recursive formula represents Newton's method.

Newton's method

- There is no guarantee that Newton's algorithm heads in the direction of decreasing values of the objective function even if $H(\boldsymbol{x}^{(k)}) \succ 0.$

- Thus, the Newton's method can be modified

  as follows

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k H(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$$

where $\alpha_k = \arg\min_{\alpha \geq 0} \; f(x^{(k)} - \alpha H(x^{(k)})^{-1} \nabla f(x^{(k)}))$

- Then the modified Newton's method is guaranteed to decrease, that is, $f(x^{(k+1)}) < f(x^{(k)})$ as far as $H(x^{(k)}) \succ 0$ and $\nabla f(x^{(k)}) \neq 0$.

Newton's method

- Newton's method has superior convergence properties when the starting point is near the solution.
- Drawbacks
  - 1) Evaluation of $H(x^{(k)})$ for large systems (i.e., large n) can be computationally expensive.
  - Furthermore, we have to solve the set of n linear equations

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - H(\boldsymbol{x}^{(k)})^{-1}\nabla f(\boldsymbol{x}^{(k)}) \qquad \text{or}$$

$$H(\boldsymbol{x}^{(k)})\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)}), \quad \text{where} \quad \boldsymbol{d}^{(k)} = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}.$$

$\Rightarrow$ Use the Quasi-Newton method (approximating the inverse Hessian)

  - 2) Hessian matrix may not be positive definite. (next slide)

Newton's method

- Drawback 2) what if H is not positive definite?
  - The search direction $d^{(k)} = -H(x^{(k)})^{-1}\nabla f(x^{(k)})$ may not point in a decent direction.

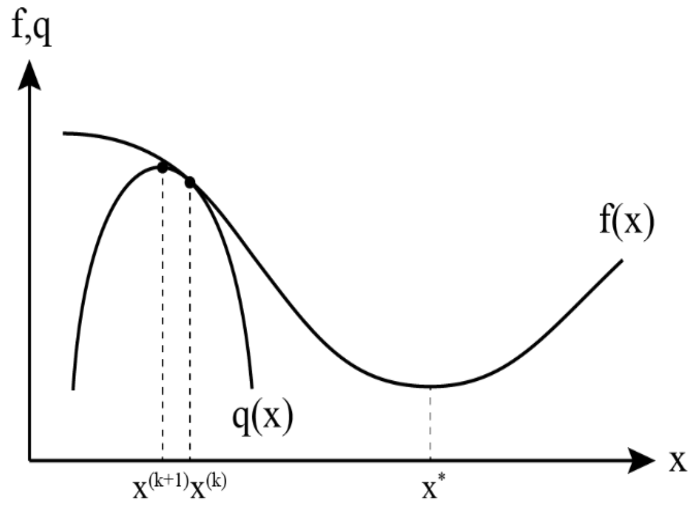
- Solution is to use Lavenberg-Marquardt(LM) modification to ensure a descent direction.

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - (H(\boldsymbol{x}^{(k)}) + \mu_k I)^{-1}\nabla f(\boldsymbol{x}^{(k)}), \quad \mu_k > 0.$$

  - Considering $L = H + \mu I$
  - Then,

$$\begin{aligned} Lv_i &= (H + \mu I)v_i \\ &= Hv_i + \mu I v_i \\ &= \lambda_i v_i + \mu v_i \\ &= (\lambda_i + \mu)v_i \end{aligned}$$

Therefore, if μ is sufficiently large, then all the eigenvalues of L are positive and L is positive definite.



Newton's method

- Finally, along with the step size α_k,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \left(\boldsymbol{H}\left(\boldsymbol{x}^{(k)}\right) + \mu_k \boldsymbol{I}\right)^{-1} \nabla f\left(\boldsymbol{x}^{(k)}\right)$$

- Now we are guaranteed that the descent property holds.

Newton's Method for Nonlinear Lease Square problems

KAIST

- Particular class of optimization problems and the use of Newton's method

$$\textit{\textbf{minimize}} \sum_{i=1}^{N} r_i(\boldsymbol{x})^2, \quad r_i : R^n \to R$$

- Ex) Fitting sinusoid to the measurement data (i.e., regression problem of measured N points)

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} r_i(\boldsymbol{x})^2 = \sum_{i=1}^{N} (y_i - A\sin(wt_i + \phi))^2$$

  – The goal is to find $\boldsymbol{x} = [A, w, \phi]^T$ that minimizes the objective function f(x) by using Newton's method.

Newton's Method for Nonlinear Lease Square problems

- Define a vector function $\boldsymbol{r} = [r_1, \ldots, r_N]$, then $f(\boldsymbol{x}) = \boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$.
- Take derivative with regard to x_j

$$\nabla f_j(\boldsymbol{x}) = \frac{\partial f}{\partial x_j}(\boldsymbol{x}) = 2\sum_{i=1}^{N} r_i(\boldsymbol{x}) \frac{\partial r_i}{\partial x_j}(\boldsymbol{x})$$

- Also denote the Jacobian matrix of $\boldsymbol{r}$ by

$$J_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial r_1}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & \vdots \\ \frac{\partial r_N}{\partial x_1}(\boldsymbol{x}) & \cdots & \frac{\partial r_N}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}$$

- Then the gradient of f can be represented as

$$\nabla f(x) = 2\boldsymbol{J}_r(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$$

Newton's Method for Nonlinear Lease Square problems

- The elements the gradient of *f* can be expressed as

$$\nabla f_j(\boldsymbol{x}) = 2\sum_{i=1}^{N} r_i(\boldsymbol{x}) J_{ij}(\boldsymbol{x}) \quad \text{with} \quad J_{ij}(\boldsymbol{x}) = \frac{\partial r_i(\boldsymbol{x})}{\partial \boldsymbol{x}_j}.$$

- Hessian Matrix $H_{jl}(\boldsymbol{x}) = \dfrac{\partial \nabla f_j(\boldsymbol{x})}{\partial x_l} = 2\sum_{i=1}^{N}\left(\dfrac{\partial r_i(\boldsymbol{x})}{\partial x_l} J_{ij}(\boldsymbol{x}) + r_i(\boldsymbol{x}) \dfrac{\partial J_{ij}(\boldsymbol{x})}{\partial x_j}\right)$

$$= 2\sum_{i=1}^{N}\left(J_{il}(\boldsymbol{x}) J_{ij}(\boldsymbol{x}) + r_i(\boldsymbol{x}) \frac{\partial^2 r_i(\boldsymbol{x})}{\partial x_j \partial x_l}\right)$$

$$\boldsymbol{H}(\boldsymbol{x}) = 2(\boldsymbol{J}_r(\boldsymbol{x})^T \boldsymbol{J}_r(\boldsymbol{x}) + \boldsymbol{S}(\boldsymbol{x}))$$

S(x) is negligibly small

- Therefore, Newton's method applied to the nonlinear LS problem is

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - (\boldsymbol{J}_r(\boldsymbol{x})^T \boldsymbol{J}_r(\boldsymbol{x}))^{-1} \boldsymbol{J}_r(\boldsymbol{x})^T \boldsymbol{r}(\boldsymbol{x})$$

← Gauss-Newton method

Levenberg-Marquardt Algorithm

- Since $J_r(x)^T J_r(x)$ may not be positive definite, LM modification is used;

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}_r(\mathbf{x})^T \mathbf{J}_r(\mathbf{x}) + \mu_k \mathbf{I})^{-1} \mathbf{J}_r(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$ ←Levenberg-Marquardt algorithm

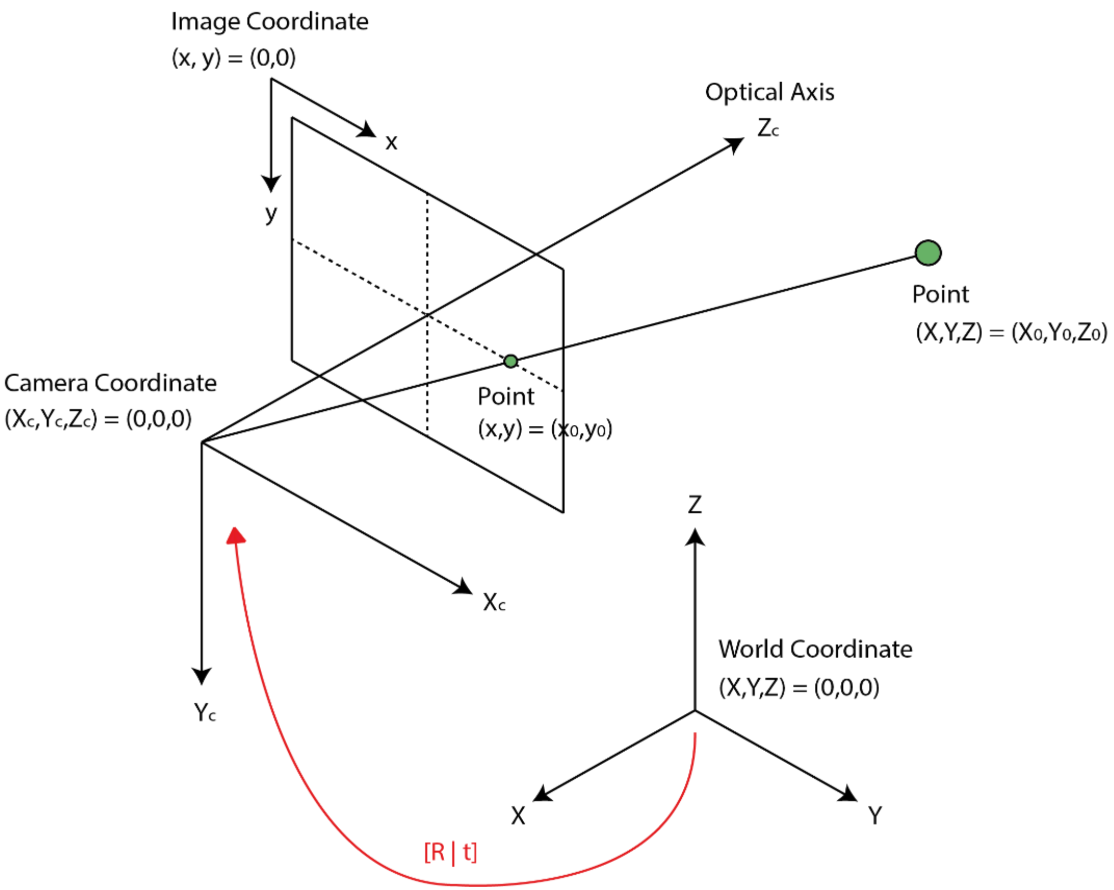
- The variant proposed by Marquadt (1963)

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}_r(\mathbf{x})^T \mathbf{J}_r(\mathbf{x}) + \mu_k \cdot diag\,(\mathbf{J}_r(\mathbf{x})^T \mathbf{J}_r(\mathbf{x}))^{-1} \mathbf{J}_r(\mathbf{x})^T \mathbf{r}(\mathbf{x})$$

  – Using the diagonal of Hessian instead of the identity allows one to take into account information about the curvature even when μ_k is large.

Levenberg-Marquardt Algorithm

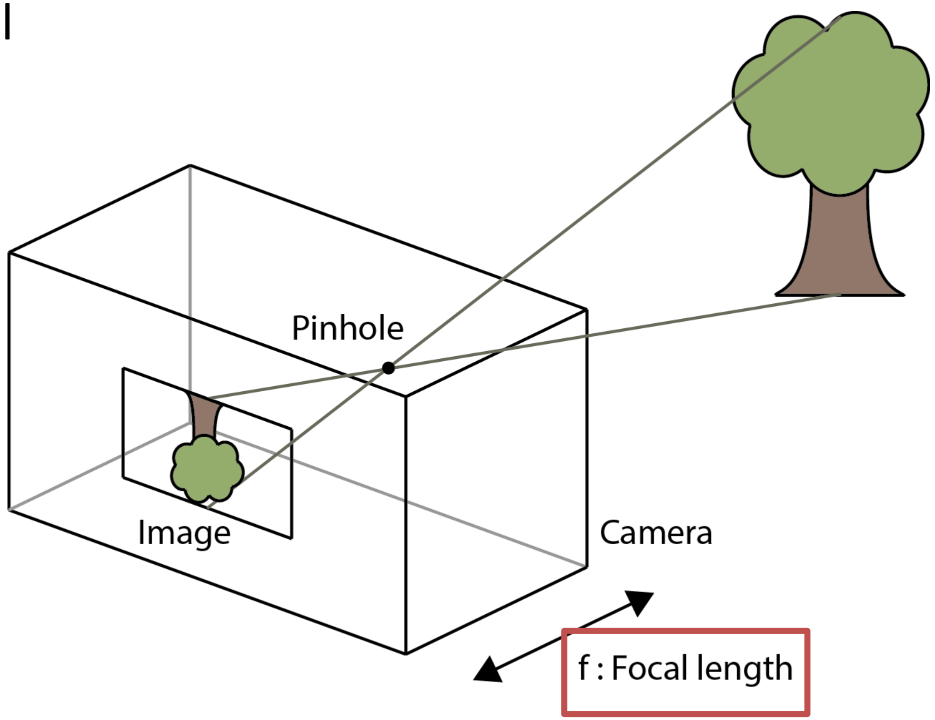
- Example 2: Camera Calibration [Zhang, PAMI 2000]
  - Pinhole camera model
    - 3D object -> 2D Image
    - Three coordinates
      - World(X,Y,Z)
      - Camera(Xc,Yc,Zc)
      - Image(x,y)
    - Intrinsic parameter (A)
      - Camera <-> Image
    - Extrinsic parameter ([R|t])
      - World <-> Camera



<Pinhole camera model>

Levenberg-Marquardt Algorithm

- Example 2: Camera Calibration [Zhang, PAMI 2000]
  - Introduction
    - Pinhole camera model



<Pinhole camera model>

Levenberg-Marquardt Algorithm

- Example 2: Camera Calibration [Zhang, PAMI 2000]
  - Introduction
    - Example of pinhole camera model
      - Remind the matrix

World coordinate

Scalar

$$\underbrace{s}_{\text{Scalar}}\underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\text{Image coordinate}} = \underbrace{\begin{bmatrix} f_x & alpha\_c & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic parameter}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{\text{Extrinsic parameter}} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}}_{\text{World coordinate}}$$

Image coordinate

Intrinsic parameter
(Camera coordinate -> Image coordinate)

Extrinsic parameter
(World coordinate -> Camera coordinate)

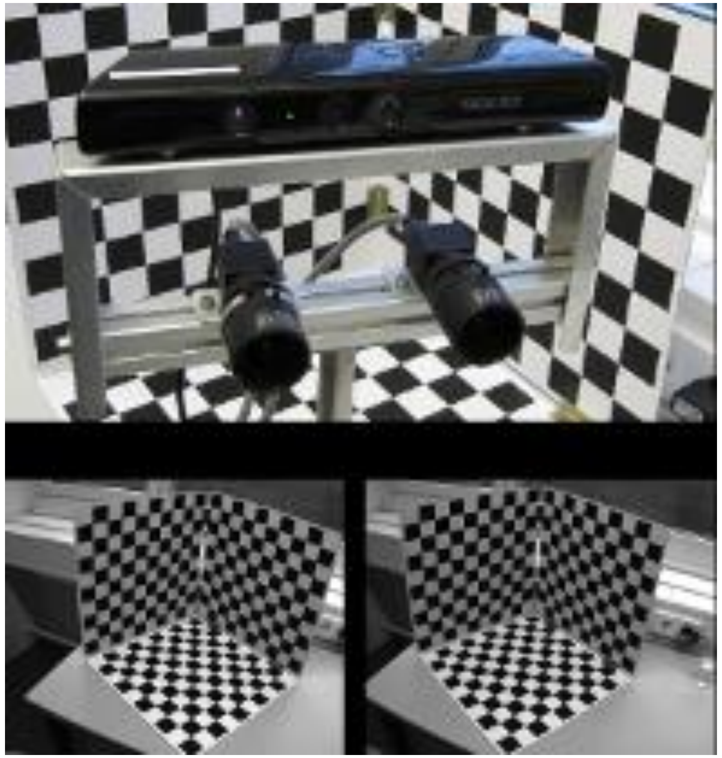
$$\text{Image} \xleftrightarrow{\text{Intrinsic parameter}} \text{Camera} \xleftrightarrow{\text{Extrinsic parameter}} \text{World}$$

Levenberg-Marquardt Algorithm

- Example 2: Camera Calibration [Zhang, PAMI 2000]
  - Optimization by LM method

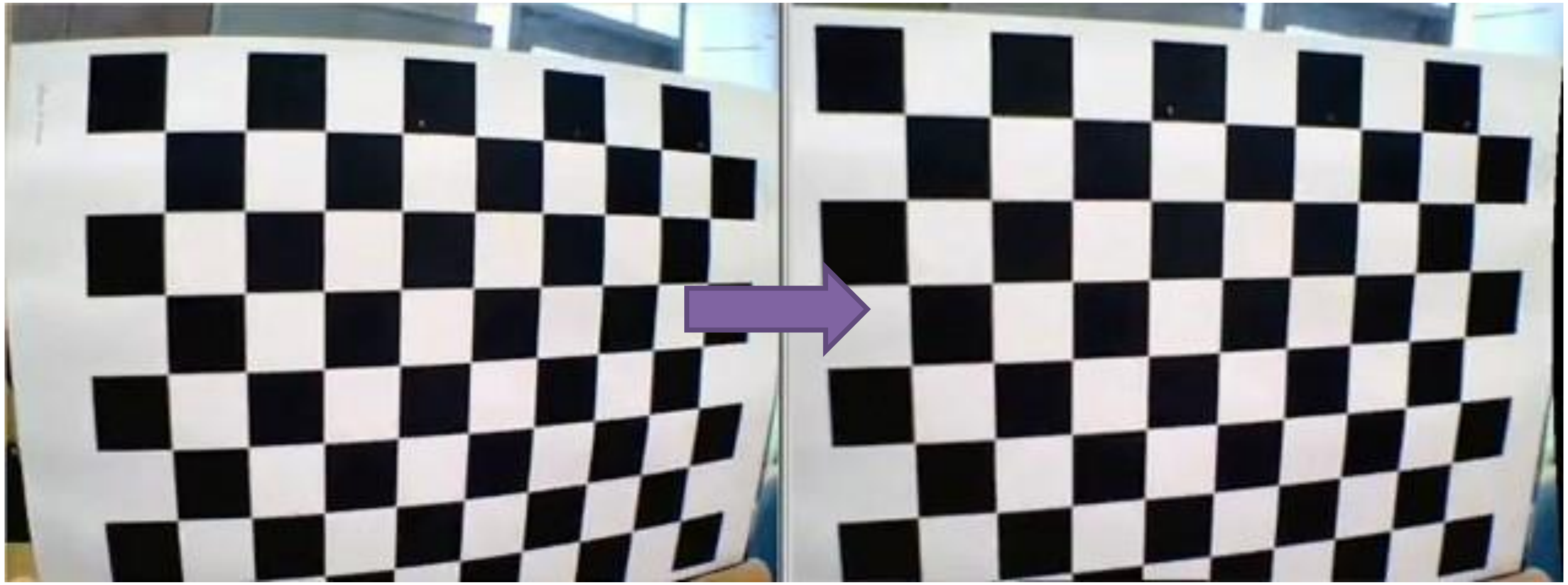
$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{i,j} - \hat{\mathbf{m}}(\mathbf{A},\mathbf{R}_i,\mathbf{t}_i,\mathbf{M}_j)\right\|^2$$

$\hat{\mathbf{m}}(\mathbf{A},\mathbf{R}_i,\mathbf{t}_i,\mathbf{M}_j)$ is the projection of point M_j in image i



Levenberg-Marquardt Algorithm

- Example 2: Camera Calibration [Zhang, PAMI 2000]
  - Experiments
    - Calibration results (Undistort image)



<Image before calibration> <Image after calibration>

References

- Camera Calibration Toolbox for Matlab : http://www.vision.caltech.edu/bouguetj/calib_doc/
- Distortion : https://photographylife.com/what-is-distortion
- Intrinsic parameter : http://ksimek.github.io/2013/08/13/intrinsic/
- Flexible Camera Calibration by Viewing a Plane from Unknown Orientations - Zhang, ICCV99
- A Four-step Camera Calibration Procedure with Implicit Image Correction - Heikkil and Silven, CVPR97
- Zhang, “A flexible new technique for camera calibration,” *IEEE PAMI, vol22, no 11, Nov. 2000*.

References

- Chong, An introduction to optimization, 3rd Ed., Wiley
- Optimization for computer vision, by Treiber, Springer
- Computer vision, by Prince, Cambridge
- www.wikipedia.com